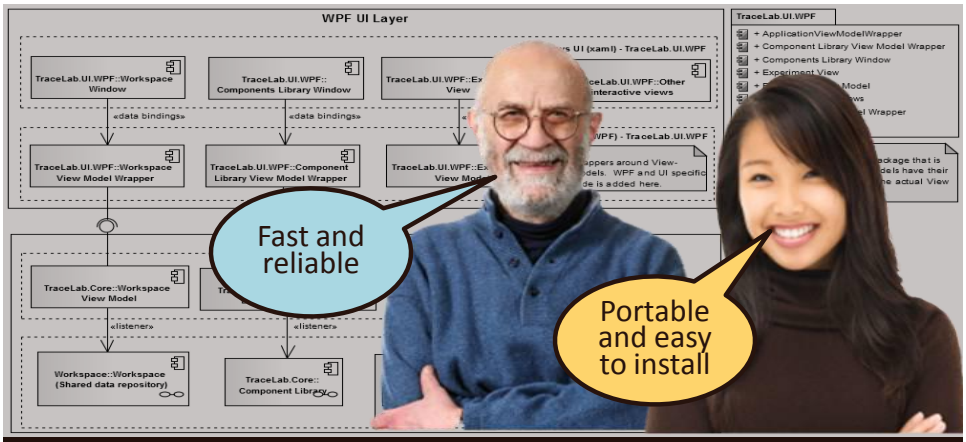


PRESERVING, GENERATING, AND VISUALIZING KNOWLEDGE OF ARCHITECTURALLY SIGNIFICANT REQUIREMENTS IN SOURCE CODE

Institute for Software Research
Distinguished Speaker Series
University of California, Irvine
April 19th, 2013
Dr. Jane Cleland-Huang
DePaul University

Research funded by the US National Science Foundation under Grants CCF-0959924 and CCF-1265178.

Architecturally Significant Requirements

- Play a strategic role in driving architectural design
- Often critical to the success (or failure of a system).
- Often represent quality concerns such as performance, portability, reliability etc.
- **Non-functional Requirements** (NFRs) are often overlooked in the requirements specification process.

Example: A medical device used to perform laser surgery must be **highly responsive.**

2

Talk Outline

- Architecturally Significant Requirements and their impact on architectural design.
  - Focus on agile projects
  - Examples from TraceLab project
- Establishing and utilizing trace links between quality concerns and code
  - Patterns of traceability
  - Archie tool
- Recovering architectural knowledge
  - Machine learning techniques

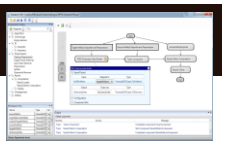
3

Working with ASRs

- In practice ASRs (especially NFRs) are often not elicited and are not clearly specified.
  - Many Software Requirements Specifications simply don't include NFRs.
  - Similarly, many agile projects fail to include ASR-related user stories.
- **Is there a better way**?
- In our TraceLab project we adopted a **persona-driven approach** which enabled us to **discover architecturally significant requirements** early in the project and to use our knowledge to make **informed decisions** about architectural design and implementation.

4

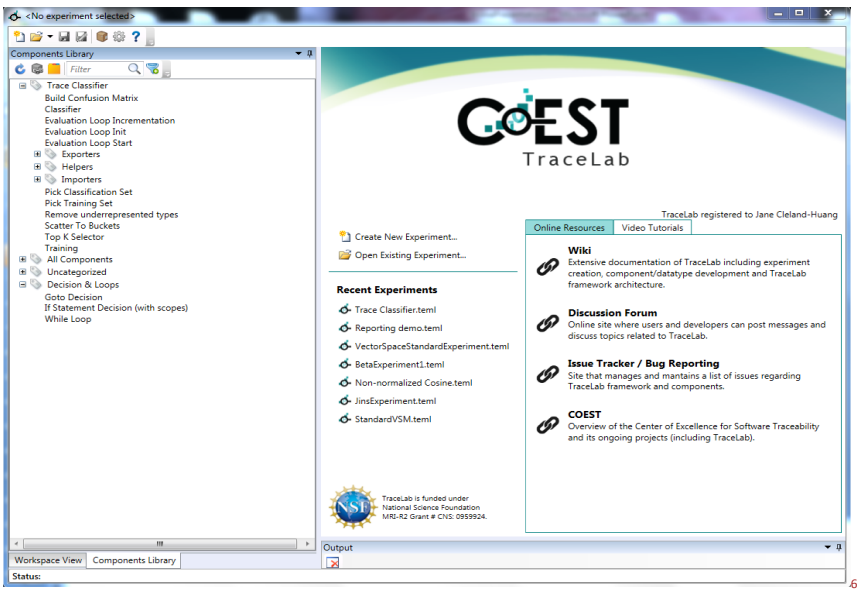
ASRs in TraceLab



- TraceLab is a US $2 Million Project funded by the National Science Foundation
- Developed by collaborators at DePaul University, College of William and Mary, Kent State Univ., and Univ. of Kentucky.
- Intended to **empower future traceability research** through facilitating innovation and creativity, increasing collaboration between traceability researchers, decreasing the startup costs and effort of new traceability research projects, and fostering technology transfer.
- Provides an environment in which researchers can **design and execute experiments**, **share components and datasets**, and **comparatively evaluate results** in a controlled setting.

5

ASRs in TraceLab



6

Competing Tradeoffs

We want to write components in C#.

We're programming this thing and we say that we should just program for Windows and everyone will have to use Windows.

I only know java. I'm not learning another language.

It's going to work on Mac, right?

It bet… fast as …periments …write myself.

I don't want to do any programming.

I want to i… my P…

We need to cut across the chaos.

We should get this into the hands of our users early so we can get early feedback.

Our lab does everything on Linux.

I'm … …ther …components …ess I know they are going to work.

I just want it to install and run easily.

No …

Can you offer it as a service?

I need to be able to write my own components.

I'm willing to share with others, but not until after I've published.

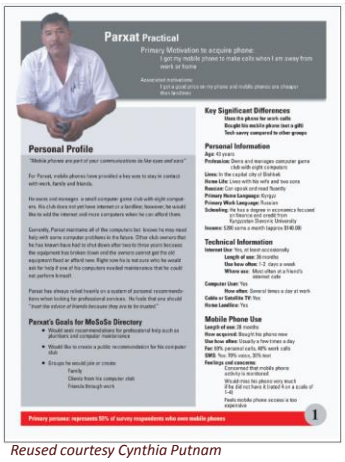
I have to run it on my desktop as I have proprietary data.

We only have 3 years to deliver everything!!

7

Traditional HCI Personas

We decided to represent the conflicting needs through developing a set of architecturally-savvy personas.



Reused courtesy Cynthia Putnam

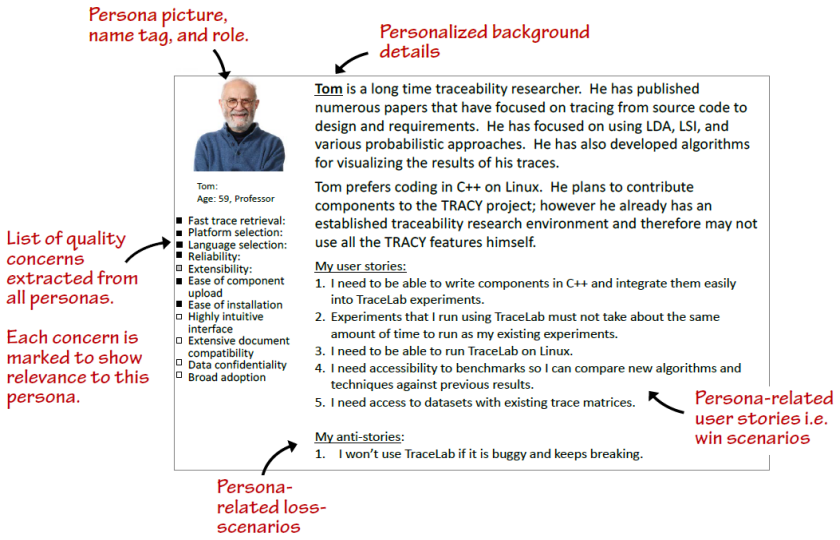
Traditionally persona construction involves surveying users, classifying them, formulating hypotheses of use, validating, creating scenarios, and finally designing personas.

Too time consuming for our project i.e. too much upfront effort that would retard the achievement of our goals.

Solution: **Persona sketches.**

8

Architecturally-Savvy Personas (Lite)



Jane Cleland-Huang, Adam Czauderna, Ed Keenan: A Persona-Based Approach for Exploring Architecturally Significant Requirements in Agile Projects. REFSQ 2013: 18-33

9

Meet Karly…

Karly
Age: 26, PhD Student

- ☐ Fast trace retrieval:
- ■ Platform selection:
- ■ Language selection:
- ☐ Reliability:
- ■ Extensibility:
- ☐ Ease of component upload
- ☐ Ease of installation
- ☐ Highly intuitive interface
- ☐ Extensive document compatibility
- ☐ Data confidentiality
- ☐ Broad adoption

Karly is a new PhD student. She is interested in tracing requirements to software architecture.

She has contacts with a local company who will allow her to access their data for her experiments; however this data is proprietary (i.e. protected by a NDA) and so she cannot share it with anyone else.

She predicts that it will take her about 6 months to set up her traceability environment, but then she discovers TRACY. Karly is quite a good programmer, but is much more interested in the process side of her research.

My user stories:

1. I need to be able to maintain confidentiality of my data.
2. I need to be able to create my own components and integrate them with existing experiments.
3. I need to be able to setup new benchmarks for comparative purposes.
4. I need to be able to program components in C#.

10

Meet Jack..

Jack, 34
Architect

- ☐ Fast trace retrieval:
- ■ Platform selection:
- ■ Language selection:
- ☐ Reliability:
- ☐ Extensibility:
- ☐ Ease of component upload
- ☐ Ease of installation
- ☐ Highly intuitive interface
- ☐ Extensive document compatibility
- ☐ Data confidentiality
  Broad adoption

Jack is married and has two young children. He has recently been hired by the TRACY project into the role of Software Architect/Developer. He has 6 years of experience as a software developer and 2 years as a lead architect in a successful gaming company. He has taken the job on the TRACY project because he is excited by the challenge of working in a research oriented project.

Jack is very motivated to build a high quality product. Jack has never worked in an academic research setting before. He is very collaborative and is looking forward to working with the other developers, academics, and students on the project.

My user stories:

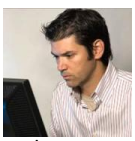
1. I need to develop the TraceLab framework in a language which supports rapid prototyping.
2. I need the framework language to easily interface with, and call, components written in other languages.
3. I need the platform to provide natural support for the separation of model and view components.
4. I need libraries to support GUI development.

11

Meet the full ensemble…

Tom

Karly



Jack

Glen
Age: 23
MS Student at Hillsbury College

Glen is an MS student who has been helping his advisor to build TraceLab components. He has never contributed to an open source project before, so he needs to figure out how to make contributions to TraceLab. Glen is very collaborative and is looking forward to working with the other researchers on the project.

Wayne
Age:46
Technical Project Mgr
ABC Corp

Wayne is the technical manager for a very large systems engineering project. He could be described as an early adopter, as he prides himself in keeping an eye out for good ideas that could help his organization. Wayne wants to improve the efficiency of traceability practices in his organization and is interested in using TraceLab.

Mary
Age: 51
NSF Program Officer

Mary is the funding officer for the grant. She is concerned that the project delivers on time and ultimately meets all major goals in terms of adoption, research advancements, and technology transfer.

12

Understand key concerns

Decision:		Platform/Language	 Tom	 Janet	 Karly	 Jack	 Mary	 Wayne
Pertinent user stories:	US 1.	The system must run on multiple platforms	●	●	●		●	
	US 2.	Users must be able to write and integrate components from multiple languages	●	●	●		●	
	US 3.	The source language of each component must be invisible at runtime				●		
	US 4.	The selected language/platform must support rapid framework prototyping				●		
	US 5.	The selected GUI must deliver 'razzle dazzle'		●		●		●
Architectural Decisions	AD 1.	Build framework using Visual Studio.net and C#.	½	✓	✓	✓	½	✓
	AD 2.	Develop the initial Windows-specific GUI in WPF.						
	AD 3.	Utilize MVVM (model view view model) architectural pattern, so that (a) the GUI View is loosely coupled and can be later implemented using GTK or Windows Forms and compiled for multiple platforms, and (b) the TraceLab engine can be compiled using Mono for porting to Linux and Mac environments.						

Process steps:

1. Analyze persona needs.
2. Identify primary drivers.
3. Extract all related user stories.
4. Assign to personas.
5. Brainstorm architectural design solutions and evaluate leading contenders.
6. Evaluate against personas.

13

Design solutions for key concerns

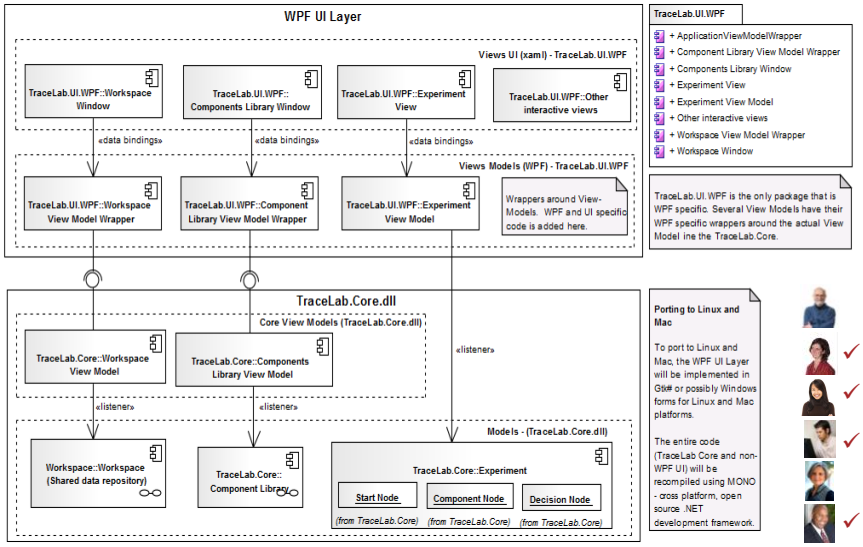
Decision:		Platform/Language	 Tom	 Janet	 Karly	 Jack	 Mary	 Wayne
Architectural Decisions	AD 1.	Build framework using Visual Studio.net and C#.	½	✓	✓	✓	½	✓
	AD 2.	Develop the initial Windows-specific GUI in WPF.						
	AD 3.	Utilize MVVM (model view view model) architectural pattern, so that (a) the GUI View is loosely coupled and can be later implemented using GTK or Windows Forms and compiled for multiple platforms, and (b) the TraceLab engine can be compiled using Mono for porting to Linux and Mac environments.						
Risks	R 1.	The Mono library may not support latest features of C#. Better support for Linux than Mac.	Long running OS project. Initial tests showed adequate support. Mitigate risk through frequent Mono compiles throughout the project.					
	R 2.	Build first for Windows solution may lead to multiple GUIs to maintain in the long run.	Decision is deferred as to whether the WPF version will be maintained or discarded in favor of a multi-platform GUI over the long term.					
Personal Impacts	PI 1.	Tom & Mary's needs are partially met through this solution. In the long-term researchers will be able to use TraceLab in Linux, but early releases will run on Windows only.						
	PI 2.	All other personas impacted directly by platform/language decisions are positively impacted by this decision.						

Process steps:

7. Identify architectural risks associated with the proposed solution and their mitigations.
8. Consider and document impacts upon personas.

14

Architectural design



Supports build-now/port-later decision

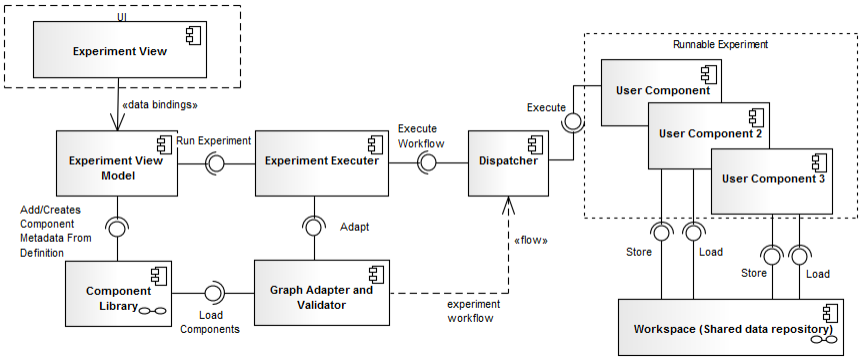
Decision 2: Workflow architecture

Decision:		Workflow Architecture	Tom	Janet	Karly	Jack	Mary	Wayne
Pertinent user stories:	US 1.	The TraceLab environment must support plug and play.	●	●	●			●
	US 2.	The performance penalty of using TraceLab must be low (i.e. close to runtime of non-TraceLab experiments).	●	●	●			●
	US 3.	Components should be reusable across research groups and experiments.	●				●	
Architectural Decisions	AD 1.	Utilize a blackboard architecture.	½	✓	✓			✓
	AD 2.	Create standard data types for exchanging data between components.						
	AD 3.	Construct the experiment around the concept of a workflow.						
	AD 4.	Support concurrent execution of components.						
	AD 5.	Trust the TraceLab users to create a viable workflow. Provide basic type checking only.						
Risks	R 1.	Performance may suffer as data is exchanged between components via shared memory.	Keep the data cache in the same App space as the experiment to avoid excessive data marshalling . Stream only critical data not entire data structure class.					
	R 2.	If TraceLab users proliferate the creation of data types, then plug-and-play ability will be lost.	Use community governance to increase the likelihood of shared use of data types.					
Personal Impacts	PI 1.	All personas are satisfied with the plug-and-play solution.						
	PI 2.	The performance penalty will be felt more by Tom, as he already has a functioning tracing environment. For other researchers the benefits of the plug-and-play environment and the use of previously defined tracing components far outweighs the slight performance penalty.						

Options

- Pipe-and-filter
- Services
- Precedence graph + Blackboard

Decision 2: Workflow architecture



Our approach is generalizable..

We created five Architecturally Savvy Personas for a Mechatronics Traceability project that we are working on with Siemens.

The personas highlighted different kinds of concerns from those highlighted by the TraceLab personas

Elaine, Age 50
Mechanical Engineer

- [x] Fast trace retrieval
- [x] Access control
- [ ] Extensibility to new case tools
- [ ] Interoperability of data formats
- [ ] Remote access
- [ ] Trace GUIs as plugins

Elaine is a mechanical engineer with over 20 years of experience working for Company X She is in charge of modeling the mechanisms for a railway gate. Her model needs to integrate with other models that describe the signaling process for the railway system. Elaine is aware that the crossing-gate must comply to a number of regulatory codes and she would like to be able to view the relevant codes from within her model. Elaine has access rights to update her model and to read requirements.

My user stories:

1. I need to be able to access all regulatory codes that impact the model I am currently working on.
2. I would like to control who views the models I am working on, and which version they view.
3. When I trace between my model and requirements, I need the traces to be returned within 30 seconds.
4. I need trace information to be displayed as an integral part of the model I am working in.

Stanley, Age 50
Compliance Officer

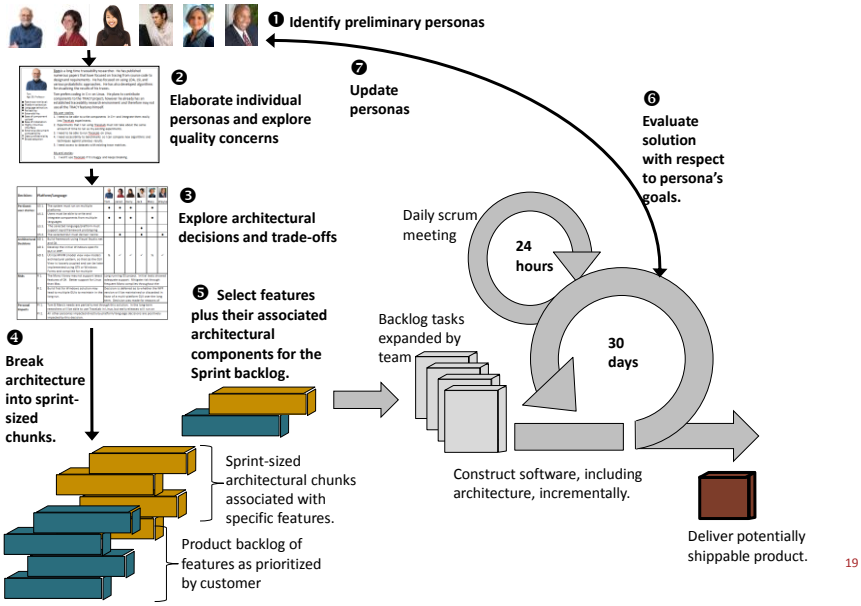
- [x] Fast trace retrieval
- [ ] Access control
- [x] Extensibility to new case tools
- [ ] Interoperability of data formats
- [ ] Remote access
- [x] Trace GUIs as plugins.

John is the compliance officer for company X. His job is to ensure that all regulatory codes are met by the delivered product and to generate reports to demonstrate this. He is a very detail-oriented person and takes great pride in his job. No products have ever been recalled under his watch for non-compliance purposes.

My user stories:

1. I need to be able to generate a report which shows a list of all elements in the design that help satisfy each relevant regulatory code. The report should generate within 2 minutes.
2. I need to view traces created in a wide variety of products.
3. I need to be able to generate and view traces for remote (i.e. globally distributed) models.

SCRUM+ ASPs



19

So what did we learn?

- Emerging and analyzing quality concerns early allowed us to make more informed architectural decisions.
- Sketching out architecturally savvy personas (ASPs) enables us to think about quality concerns in a more tangible way.
- Our approach fits naturally into the SCRUM-like process we had adopted for the project.
- A light-weight approach for integrating NFR-thinking into a fast-paced, agile, development environment.

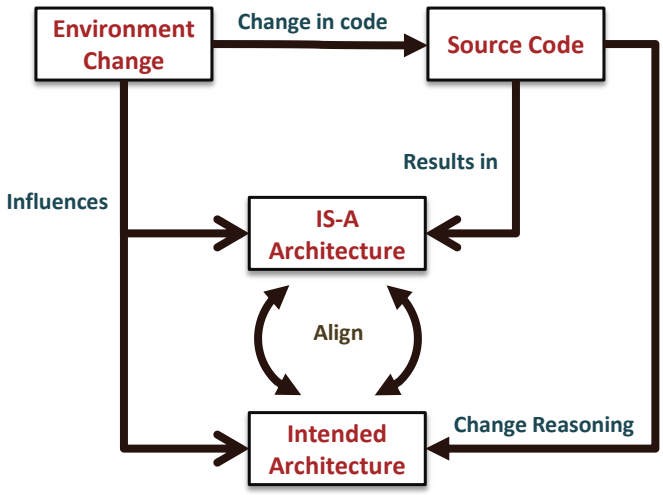
20

Talk Outline

- Architecturally Significant Requirements and their impact on architectural design.
  - Focus on agile projects
  - Examples from TraceLab project
- Establishing and utilizing traceability links between quality concerns and code
  - Patterns of traceability
  - Archie tool
- Recovering architectural knowledge
  - Machine learning techniques

21

Change Cycle: Ideal World

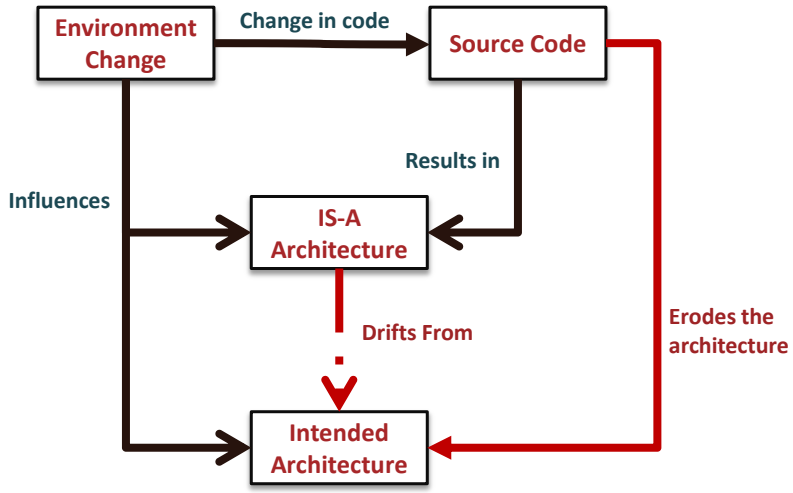


Ideal World: Architectural information is documented during the Architectural design phase and is updated regularly to reflect the current system architecture.

Slide used courtesy of Mehdi Mirakhorli

22

Change Cycle: Real World



Real World: Architectural information is outdated and does not reflect the current architecture of the system.

Slide used courtesy of Mehdi Mirakhorli

23

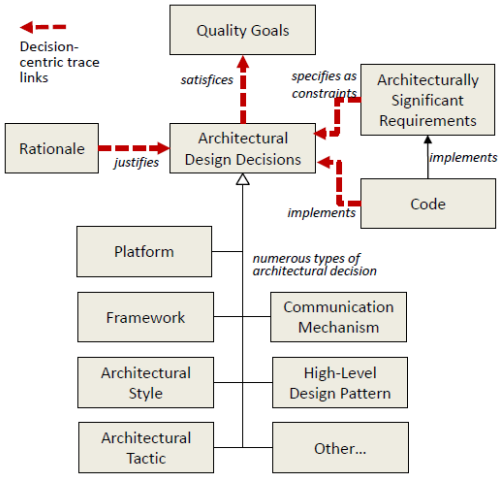
Architectural Degradation

1. Intended and implemented architecture diverge.
2. Architecture violations (i.e. strict layering bypassed, or pipe-and-filter pipeline violated); cyclic dependencies; dead code; code clones; metric outliers etc.

System becomes brittle starts to erode.

24

Tracing Concerns to Code

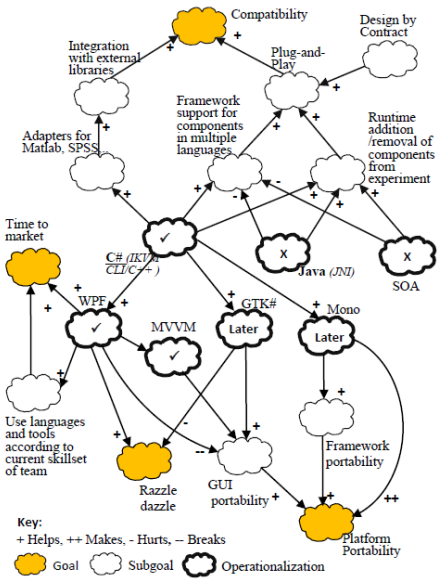


Requirements traceability is the ability to describe and **follow the life of a requirement**, in both a forward and backward direction, i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases."

Gotel and Finkelstein,1994.

25

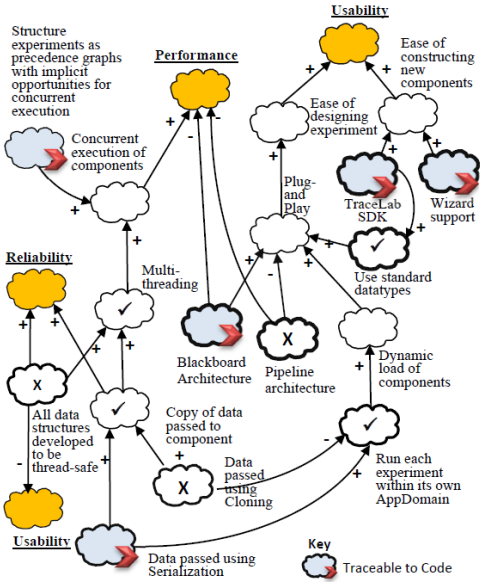
Tracing Concerns to Code



We can use the Softgoal Interdependency Graph (SIG) notation to capture the goal refinements that lead to our architectural decisions.

Decision-Centric Traceability of Architectural Concerns , Jane Cleland-Huang, Mehdi Mirakhorli, Adam Czauderna, and Mateusz Wieloch , Traceability in Emerging Forms of Software Engineering, May 2013.

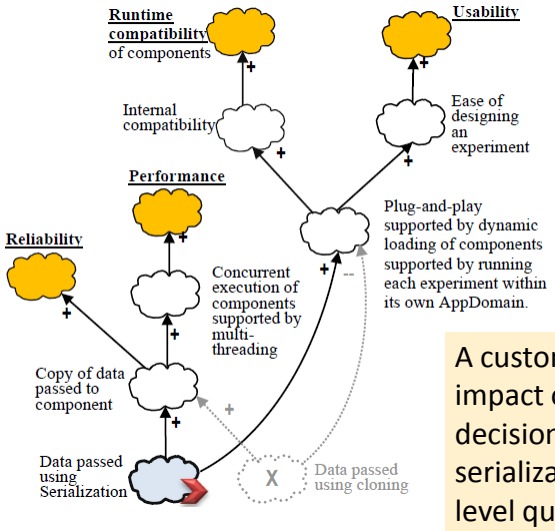
Tracing Concerns to Code



Only certain kinds of architectural decisions are traceable to code.

27

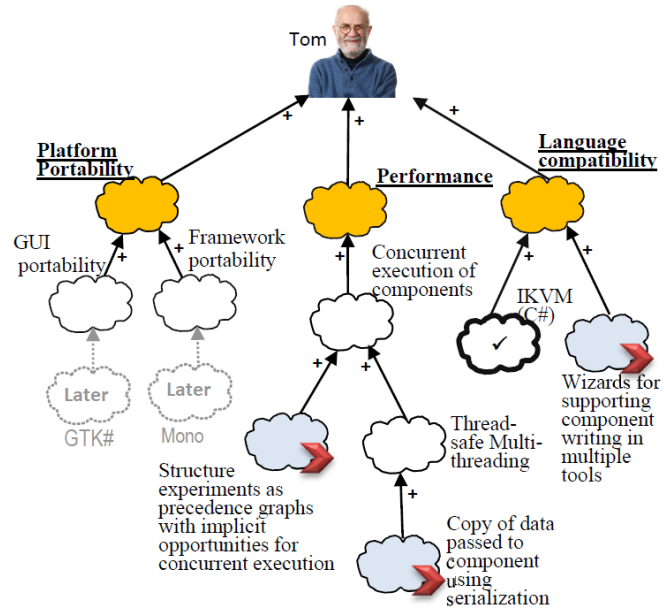
Customized Views



A custom view shows the impact of the architectural decision to pass data using serialization, on higher level quality concerns.

28

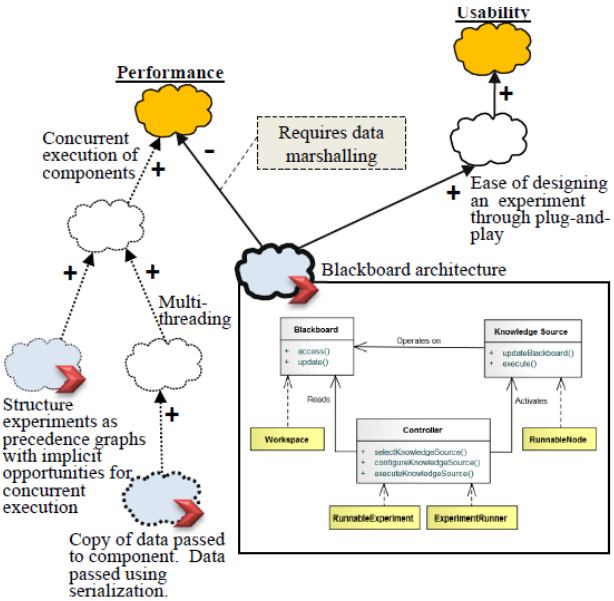
Customized Views



A persona / user perspective upon architectural decisions.

29

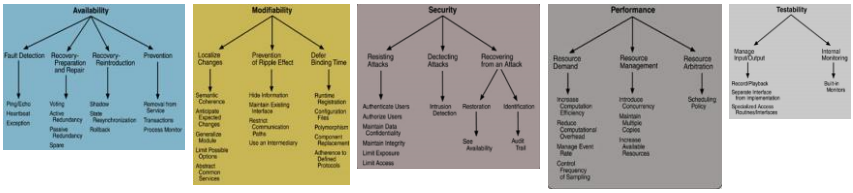
Some decisions occur across multiple projects



Can we find better ways to trace quality concerns to code when common architectural decisions are made?

30

Some decisions recur across projects



Due to complexity of the problem, we tackled tactics first.

- Tactics are pervasive in fault-tolerant and/or high-performance systems.
- Tactics seem to have an interesting relationship to change.

31

Tactic Occurrence Across Projects

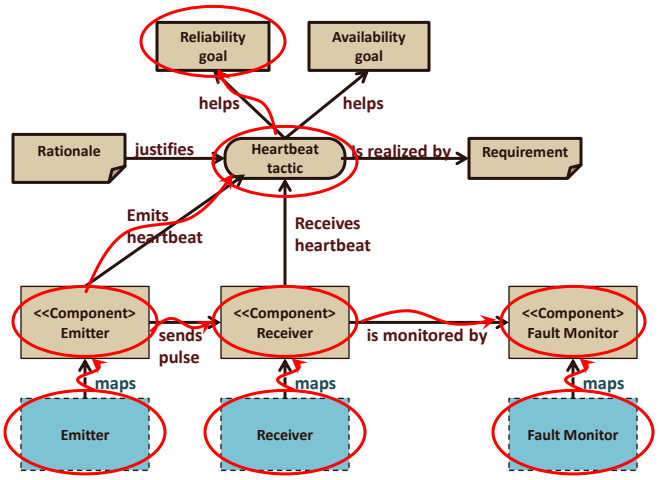
	Fault tolerant, performance-centric software systems from SourceForge	Heartbeat	Scheduling	Authentication	Audit Trail	Resource Pool.	Active Repl.	Recovery	Passive Repl.	Authorization	Permiss. Check	CRC	Encryption	Process Monitor	Rem. Service	Fault Detection	Voting
1	RIFE: a Web application engine with support for content management.		•	•		•				•			•			•	
2	Fault-Tolerant Corba: (OMG Document ptc/2000-04-04)	•	•		•	•	•	•	•					•	•	•	•
3	CARMEN: Robot Control Software, with navigation capabilities	•										•	•				
4	Rossume: an open-source robot simulator for control and navigation.	•	•			•									•	•	
5	jworkosgi: implementation of the JMX and JMX Remote API into OSGI bundles.	•	•	•		•				•	•	•	•	•			
6	SmartFrog: Distributed Application Development Framework	•	•	•	•	•		•		•	•					•	
7	CarDamom: Real-time, distributed and fault-tolerant middleware		•	•	•	•	•	•	•					•	•	•	•
8	ACLAnalyser: Tool suit to validate, verify and debug Multi Agent Systems	•	•	•	•						•					•	
9	Jfolder: Web-based application development and management tool.	•		•	•					•							
10	Enhydra shark: XPDL and BPMN Workflow Server		•	•	•	•				•	•			•			
11	Chat3: An instant messenger.	•		•									•				
12	ACE+TAO+CIAO: Framework for high-performance, distributed, real-time systems.	•	•			•	•		•			•	•			•	•
13	Google Chromium OS:	•	•	•	•	•				•	•		•			•	
14	x4technology tools: Framework Enterprise application software.			•	•	•				•		•					
15	OpenAccountingJ: web-based Accounting/ERP system.			•	•						•						
16	Airbus Family: Flight Control System*.	•	•			•	•	•			•	•			•		•
17	Boeing 777: Primary Flight Control (PFC)*.	•	•			•	•	•				•			•		•
18	NASA CEV: Crew Exploration Vehicle using guidance-navigation* & control model.	•					•	•	•			•		•	•	•	•
19	Hadoop Framework: a development framework to support cloud computing.	•	•	•	•	•				•	•	•	•			•	
20	OfBiz: an enterprise automation and E-Commerce software.		•	•	•	•				•	•		•				

Courtesy Mehdi Mirakhorli

Tactics tend to be found in safety-critical, and/or other kinds of performance-centric systems.

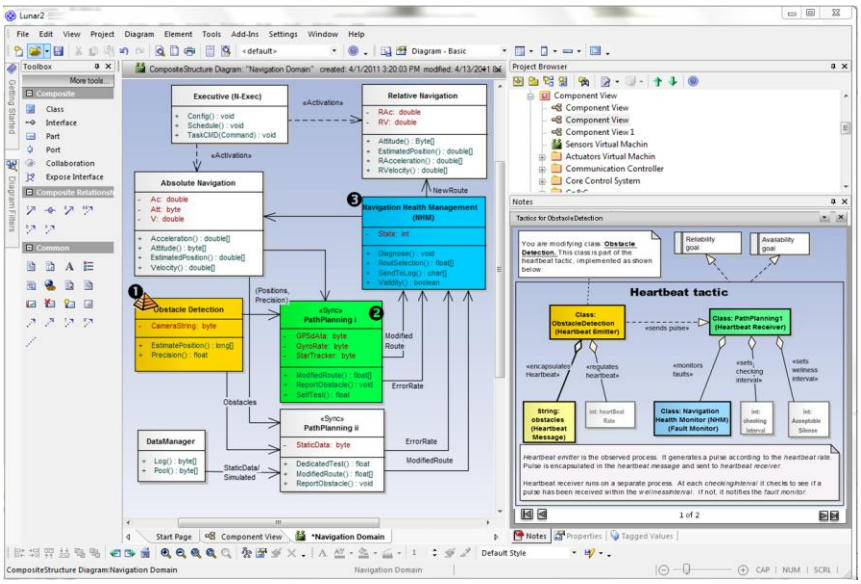
32

Tactic Traceability Patterns



- Mehdi Mirakhorli and Jane Cleland-Huang, *Using Tactic Traceability Information Models to Reduce the Risk of Architectural Degradation during System Maintenance*, International Conference on Software Maintenance, Williamsburg, USA, September, 2011
- Mehdi Mirakhorli and Jane Cleland-Huang, "A Pattern System for Tracing Architectural Concerns", *Pattern Languages of Programming*, Portland, USA, October, 2011

Archie…

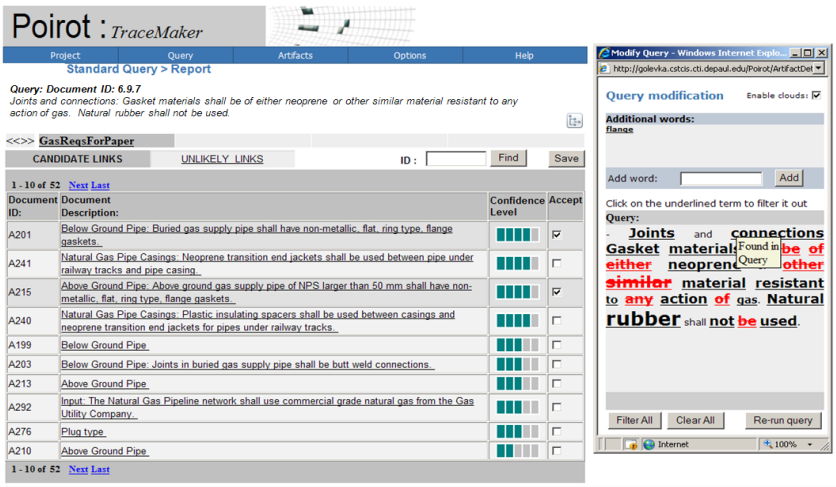


Talk Outline

- Architecturally Significant Requirements and their impact on architectural design.
  - Focus on agile projects
  - Examples from TraceLab project
- Establishing and utilizing traceability links between quality concerns and code
  - Patterns of traceability
  - Archie tool
- Recovering architectural knowledge
  - Machine learning techniques

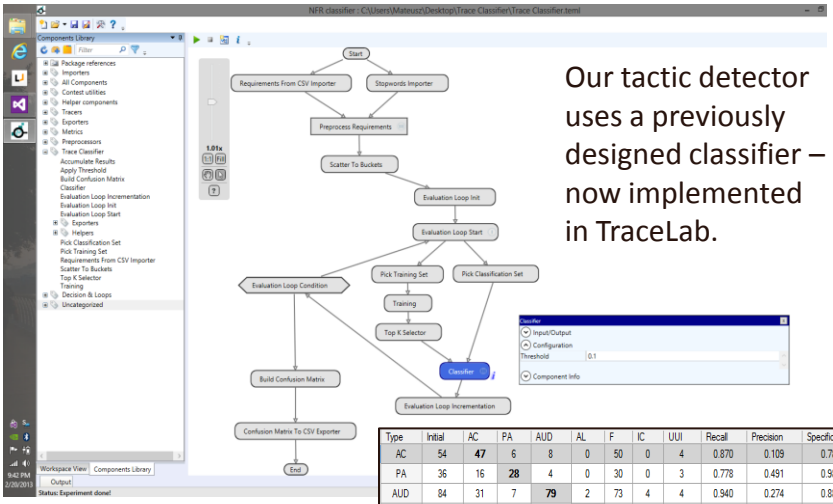
35

Trace Retrieval



In contrast, architectural concerns are often NOT unique in individual systems – so we can train our traceability engine to recognize them across projects.

Tactic Detector



Our tactic detector uses a previously designed classifier – now implemented in TraceLab.

Type	Initial	AC	PA	AUD	AL	F	IC	UUI	Recall	Precision	Specificity	F2-Measure
AC	54	**47**	6	8	0	50	0	4	0.870	0.109	0.788	0.362
PA	36	16	**28**	4	0	30	0	3	0.778	0.491	0.984	0.697
AUD	84	31	7	**79**	2	73	4	4	0.940	0.274	0.883	0.633
AL	10	5	1	2	**7**	7	0	1	0.700	0.583	0.997	0.673
F	1660	328	13	187	3	**1485**	51	67	0.895	0.891	0.142	0.894
IC	17	2	1	5	0	13	**3**	4	0.176	0.052	0.970	0.119
UUI	10	4	1	3	0	8	0	**7**	0.700	0.078	0.955	0.269

Classification Approach

① Normalizes the frequency with which term t occurs in the requirement with respect to the length of the requirement.

Computes the percentage of documents of type Q containing term t

Decreases the weight of terms that are project specific.

$$Pr_Q(t) = \frac{1}{N_Q}\sum_{r_Q \in S_Q} \frac{freq(r_Q,t)}{|r_Q|} \cdot \frac{N_Q(t)}{N(t)} \cdot \frac{NP_Q(t)}{NP_Q}$$

② $$Pr_Q(r) = \frac{\sum_{t \in r \cap I_Q} Pr_Q(r)}{\sum_{t \in I_Q} Pr_Q(t)}$$

Computes the likelihood that requirement r traces to Query q.

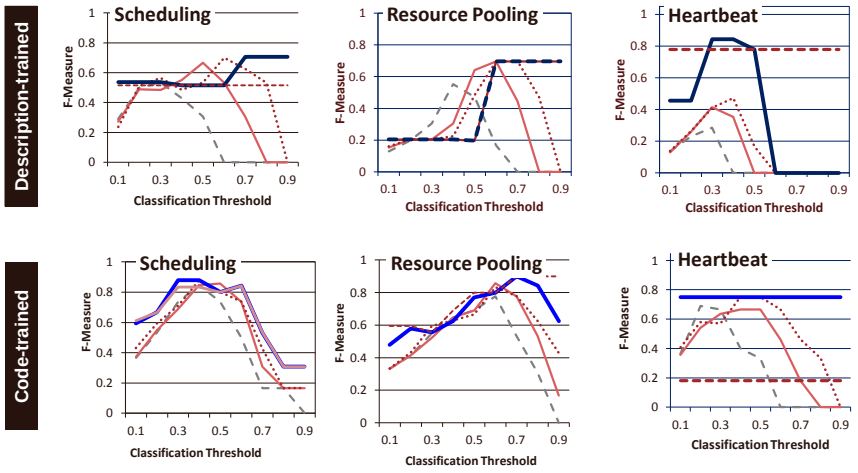
J. Cleland-Huang, R. Settimi, X.Zou, P. Solc, "Automated Detection and Classification of Quality Requirements", Requirements Engineering Journal, Springer-Verlag, August, 2006. pp. 36-45

Towards Automation

Tactic Name	Document trained indicator terms	Code trained indicator terms
Heartbeat	heartbeat, fault, detect, messag, period, watchdog, send, tactic, failur, aliv	heartbeat, ping, beat, heart, hb, outbound, puls, hsr, period, isonlin
Scheduling	prioriti, schedul, assign, process, time, queue, robin higher, weight, dispatch	schedul, task, prioriti, prcb, sched, thread, , rtp, weight, tsi
Authentication	authent, password, kerbero, sasl, ident, biometr, verifi, prove, ticket, purport	authent, credenti, challeng, kerbero, auth, login, otp, cred, share, sasl
Resource Pooling	thread, pool, number, worker, task, queue, executor, creat, overhead, min	pool, thread, connect, sparrow, nbp, processor, worker, timewait, jdbc, ti
Audit Trail	audit, trail, record, activ, log, databas, access, action, monitor, user	audit, trail, wizard, pwriter, lthread, log, string, categori, pstmt, pmr

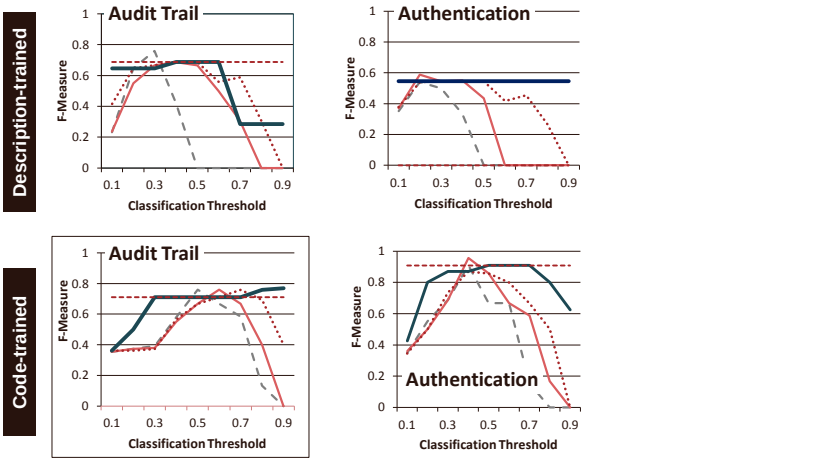
39

Tactic-Grained Classification



40

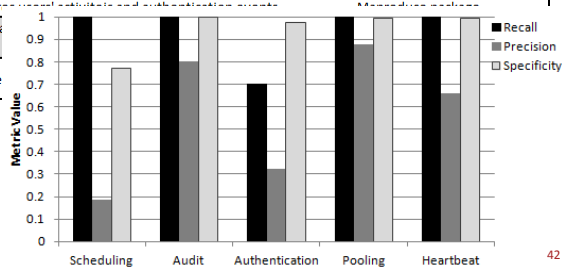
Tactic-trained Classification / Code Trained



41

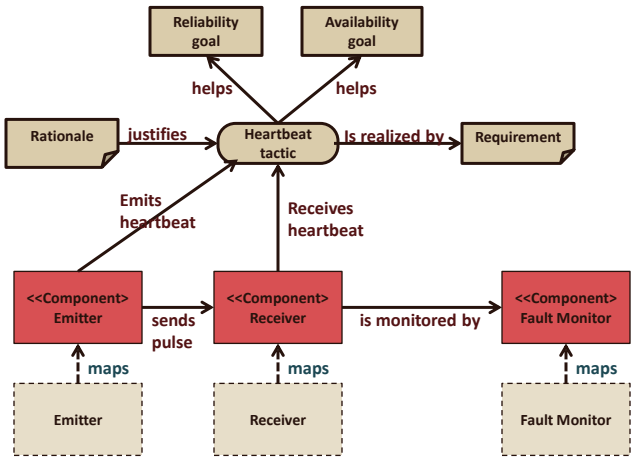
HADOOP Case Study

Tactic	Class Count	Explanation	Package name or Subsystem
Heartbeat	12	HDFS uses a master/slave architecture with replication. All slaves send a heartbeat message to the master server indicating their health	MapReduce Subsystem
	15	The MapReduce subsystem uses heartbeat with piggybacking to check the health and execution status of each task running on a cluster.	HDFS Subsystem
Resource Pooling	36	MapReduce uses Thread pooling to improve performance of many tasks e.g. to run the map function.	Mapreduce Package
	7	A global compressor/decompressor pool used to save and reuse codecs.	Compress package
	47	Block pooling is used to improve performance of the distributed file system.	HDFS subsystem
	5	Combines scheduling and job pooling. Organizes jobs into pools, and shares resources between pools.	MapReduce Subsystem
	88	Scheduling services are used to execute tasks and jobs. These include fair-, dynamic-, and capacity-scheduling.	Common and MapReduce
Scheduling	4	Audit log captur	
Audit Trail	35	Users Kerberos a subsystems.	
Authentication		The MapReduce	



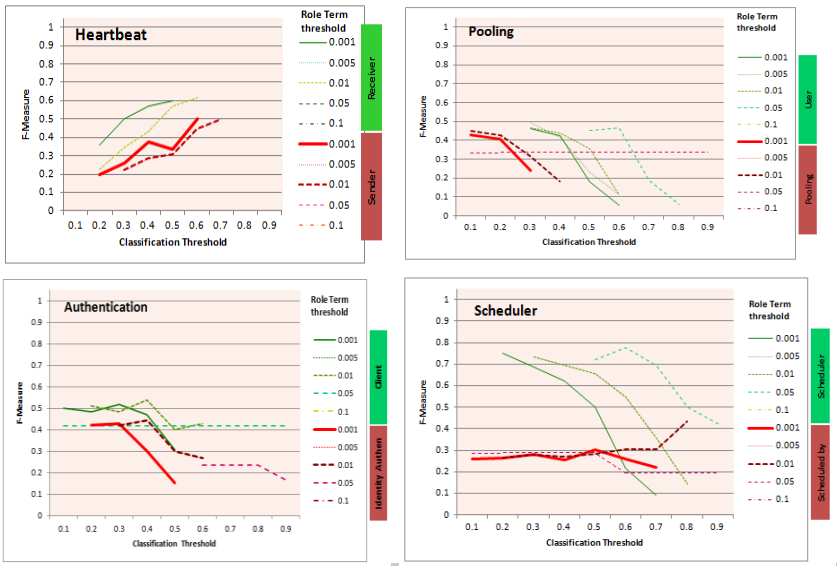
42

More Challenging: Identifying Roles



43

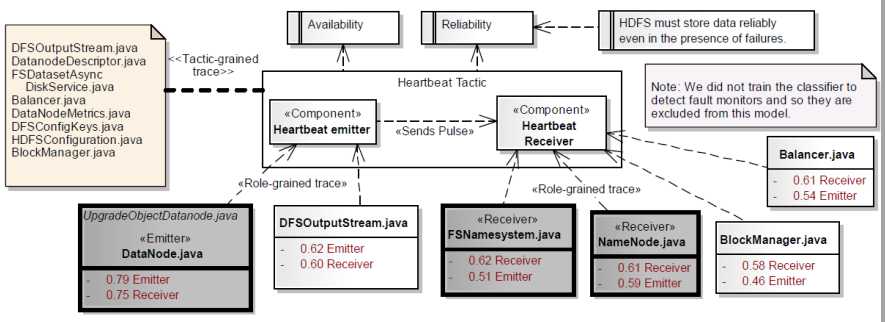
Finding Roles is Hard



We integrated light-weight structural approaches – but only evaluated them in a single case study.

44

Tactic-trained Classification / Code Trained



45

Using Generated Links to mitigate Architectural Decay

- Are automatically reconstructed traceability links good enough for use?
- Evaluated the usefulness of the generated fine-grained traceability links for supporting software maintenance.
- Utilized Hadoop change logs for the past four releases, and simulated the scenario in which generated links were used to control the generation of notification messages.

You are modifying **Datanode.java**. This file appears to play the role of **heartbeat emitter** in the heartbeat tactic.

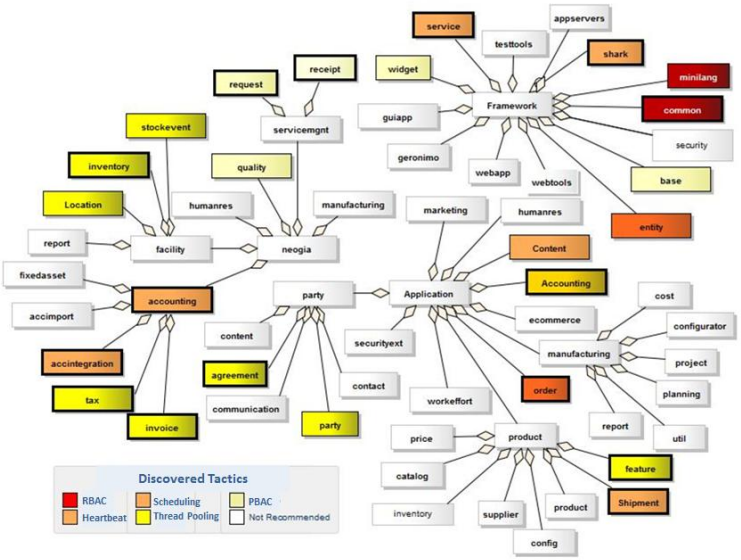
This class therefore contributes to reliability and availability goals. Tell me more.

Please confirm the role of this class in the heartbeat tactic:

- ☑ **Heartbeat emitter** (Prob 79%)
- ☐ Heartbeat sender (Prob 75%)
- ☐ Supporting role
- ☐ Unrelated to heartbeat

46

Visualizations

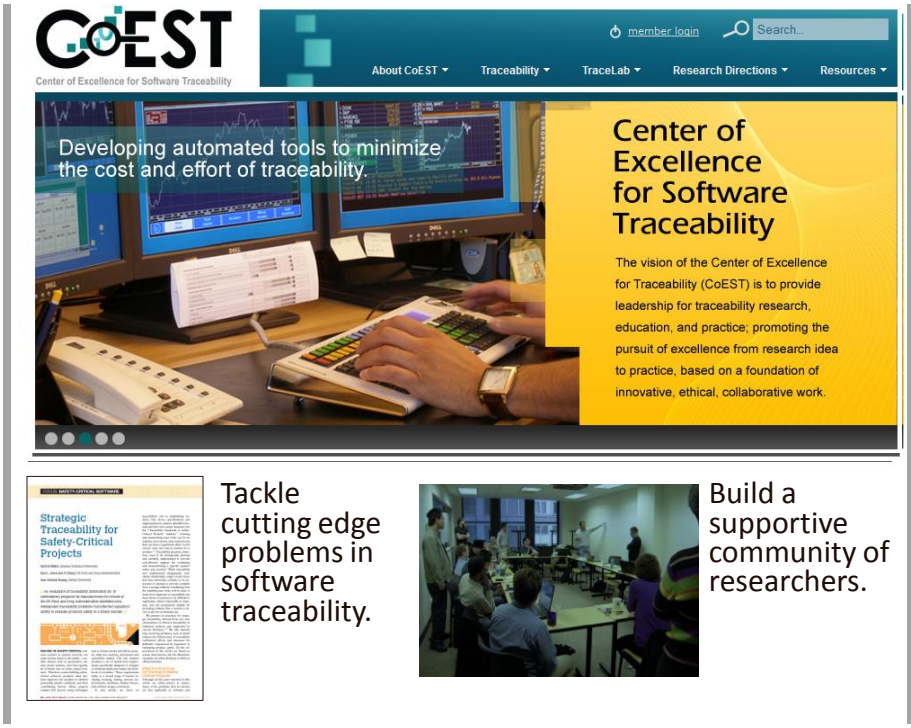


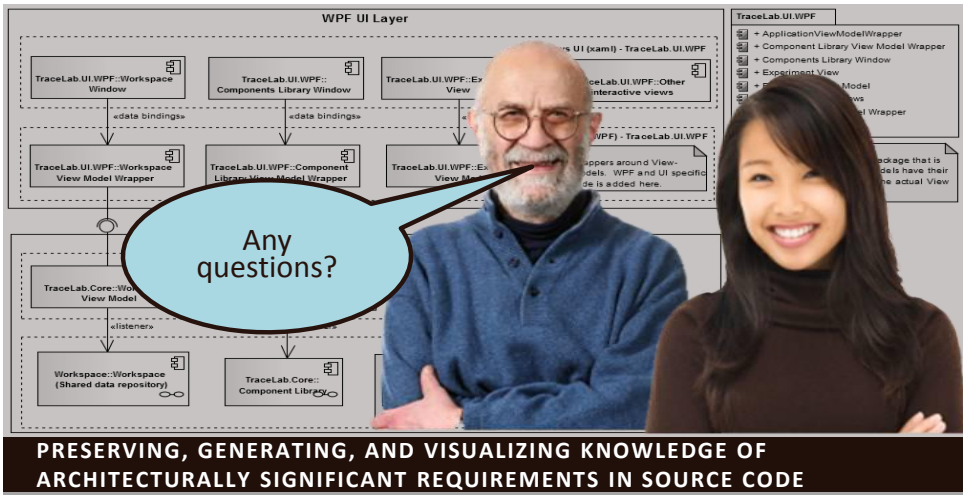
47

Conclusions

- Managing quality concerns (aka NFRs) is a complete life-cycle activity.
- Elicit them early
- Design to satisfy them
- Preserve them
- If necessary, rediscover them

48





PRESERVING, GENERATING, AND VISUALIZING KNOWLEDGE OF ARCHITECTURALLY SIGNIFICANT REQUIREMENTS IN SOURCE CODE

Institute for Software Research
Distinguished Speaker Series
University of California, Irvine
April 19th, 2013

Dr. Jane Cleland-Huang
DePaul University

Research funded by the US National Science Foundation under Major Research Instrumentation Grants CCF-0959924 and CCF-1265178.